Impact of Knowledge on Election Time in Anonymous Networks

Yoann Dieudonné[*]
Laboratoire MIS, Université de Picardie Jules Verne
33 rue Saint-Leu
Amiens 80000, France
yoann.dieudonne@u-picardie.fr

Andrzej Pelc[†]
Département d'informatique, Université du Québec en Outaouais
C.P. 1250, succ. Hull
Gatineau, Québec J8X 3X7, Canada
pelc@uqo.ca

ABSTRACT

Leader election is one of the basic problems in distributed computing. This is a symmetry breaking problem: all nodes of a network must agree on a single node, called the leader. If the nodes of the network have distinct labels, then such an agreement means that all nodes have to output the label of the elected leader. For anonymous networks, the task of leader election is formulated as follows: every node v of the network must output a simple path, which is coded as a sequence of port numbers, such that all these paths end at a common node, the leader. In this paper, we study deterministic leader election in arbitrary anonymous networks.

It is well known that deterministic leader election is impossible in some networks, regardless of the allocated amount of time, even if nodes know the map of the network. This is due to possible symmetries in it. However, even in networks in which it is possible to elect a leader knowing the map, the task may be still impossible without any knowledge, regardless of the allocated time. On the other hand, for any network in which leader election is possible knowing the map, there is a minimum time, called the *election index*, in which this can be done. Informally, the election index of a network is the minimum depth at which views of all nodes are distinct. Our aim is to establish tradeoffs between the allocated time τ and the amount of information that has to be given *a priori* to the nodes to enable leader election in time τ in all networks for which leader election in this time is at all possible. Following the framework of *algorithms with advice*, this information (a single binary string) is provided to all nodes at the start by an oracle knowing the entire network. The length of this string is called the *size of advice*. For a given time τ allocated to leader election, we give upper and lower bounds on the minimum size of advice sufficient to perform leader election in time τ.

We focus on the two sides of the time spectrum. For the smallest possible time, which is the election index of the network, we show that the minimum size of advice is linear in the size n of the network, up to polylogarithmic factors. On the other hand, we consider large values of time: larger than the diameter D by a summand, respectively, linear, polynomial, and exponential in the election index; for these values, we prove tight bounds on the minimum size of advice, up to multiplicative constants. We also show that constant advice is not sufficient for leader election in all graphs, regardless of the allocated time.

CCS CONCEPTS

• **Theory of computation** → **Distributed algorithms**; • **Mathematics of computing** → *Discrete mathematics*;

KEYWORDS

Leader election; anonymous network; advice; deterministic distributed algorithm; time

1 INTRODUCTION

Background. Leader election is one of the basic problems in distributed computing [35]. This is a symmetry breaking problem: all nodes of a network must agree on a single node, called the leader. It was first formulated in [34] in the study of local area token ring networks, where, at all times, exactly one node (the owner of a circulating token) is allowed to initiate communication. When the token is accidentally lost, a leader must be elected as the initial owner of the token.

If the nodes of the network have distinct labels, then agreeing on a single node means that all nodes output the label of the elected leader. However, in many applications, even if nodes have distinct identities, they may decide to refrain from revealing them, e.g., for privacy or security reasons. Hence it is important to design leader election algorithms that do not rely on knowing distinct labels of nodes, and that can work in anonymous networks as well. Under this scenario, agreeing on a single leader means that every node has to output a simple path (coded as a sequence of port numbers) to a common node.

Model and Problem Description. The network is modeled as a simple undirected connected n-node graph with diameter D, for $n \geq 3$. Nodes do not have any identifiers. On the other hand, we assume that, at each node v, each edge incident to v has a distinct *port number* from $\{0, \ldots, d-1\}$, where d is the degree of v. Hence, each edge has two corresponding port numbers, one at each of its endpoints. Port numbering is *local* to each node, i.e., there is no relation between port numbers at the two endpoints of an edge. Initially, each node knows only its own degree. The task of leader

[*]Partially supported by the European Regional Development Fund (ERDF) and the Hauts-de-France region under Project TOREDY.
[†]Partially supported by NSERC discovery grant 8136 – 2013, and by the Research Chair in Distributed Computing at the Université du Québec en Outaouais.

SPAA '17, July 24-26, 2017, Washington DC, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4593-4/17/07…$15.00
https://doi.org/http://dx.doi.org/10.1145/3087556.3087563

election is formulated as follows. Every node v must output a sequence $P(v) = (p_1, q_1, \ldots, p_k, q_k)$ of nonnegative integers. For each node v, let $P^*(v)$ be the path starting at v, such that port numbers p_i and q_i correspond to the i-th edge of $P^*(v)$, in the order from v to the other end of this path. All paths $P^*(v)$ must be simple paths in the graph (i.e., paths without repeated nodes) that end at a common node, called the leader. In this paper, we consider deterministic leader election algorithms.

In the absence of port numbers, there would be no way to identify the elected leader by non-leaders, as all ports, and hence all neighbors, would be indistinguishable to a node. Security and privacy reasons for not revealing node identifiers do not apply in the case of port numbers.

The central notion in the study of anonymous networks is that of the view of a node [44]. Let G be a graph and let v be a node of G. We first define, for any $l \geq 0$, the *truncated view* $\mathcal{V}^l(v)$ at depth l, by induction on l. $\mathcal{V}^0(v)$ is a tree consisting of a single node x_0. If $\mathcal{V}^l(u)$ is defined for any node u in the graph, then $\mathcal{V}^{l+1}(v)$ is the port-labeled tree rooted at x_0 and defined as follows. For every node v_i, $i = 1, \ldots, k$, adjacent to v, there is a child x_i of x_0 in $\mathcal{V}^{l+1}(v)$ such that the port number at v corresponding to edge $\{v, v_i\}$ is the same as the port number at x_0 corresponding to edge $\{x_0, x_i\}$, and the port number at v_i corresponding to edge $\{v, v_i\}$ is the same as the port number at x_i corresponding to edge $\{x_0, x_i\}$. Now node x_i, for $i = 1, \ldots, k$, becomes the root of the truncated view $\mathcal{V}^l(v_i)$.

The *view* from v is the infinite rooted tree $\mathcal{V}(v)$ with labeled ports, such that $\mathcal{V}^l(v)$ is its truncation to level l, for each l.

We use the extensively studied $\mathcal{LOCAL}$ communication model [39]. In this model, communication proceeds in synchronous rounds and all nodes start simultaneously. In each round, each node can exchange arbitrary messages with all of its neighbors and perform arbitrary local computations. The information that v gets about the graph in r rounds is precisely the truncated view $\mathcal{V}^r(v)$, together with degrees of leaves of this tree. Denote by $\mathcal{B}^r(v)$ the truncated view $\mathcal{V}^r(v)$ whose leaves are labeled by their degrees in the graph, and call it the *augmented truncated view* at depth r. If no additional knowledge is provided *a priori* to the nodes, the decisions of a node v in round r in any deterministic algorithm are a function of $\mathcal{B}^r(v)$. Note that all augmented truncated views can be canonically coded as binary strings, and hence the set of all augmented truncated views can be ordered lexicographically. The *time* of leader election is the minimum number of rounds sufficient to complete it by all nodes. It is well known that the synchronous process of the $\mathcal{LOCAL}$ model can be simulated in an asynchronous network using time-stamps.

Unlike in labeled networks, if the network is anonymous then leader election is sometimes impossible, regardless of the allocated time, even if the network is a tree and its topology is known. This is due to symmetries, and the simplest example is the two-node graph. It follows from [44] that if nodes know the map of the graph (i.e., its isomorphic copy with all port numbers indicated) then leader election is possible if and only if views of all nodes are distinct. We will call such networks *feasible* and restrict attention to them. However, even in the class of feasible networks, leader election is impossible without any *a priori* knowledge about the network. This simple observation follows from a slightly stronger result proved in Proposition 4.4. On the other hand, for any fixed feasible network G, whose map is given to the nodes, there is a minimum time, called the *election index* and denoted by $\phi(G)$, in which leader election can be performed. The election index of a network is equal to the smallest integer ℓ, such that the augmented truncated views at depth ℓ of all nodes are distinct. This will be proved formally in Section 2. The election index is always a strictly positive integer because there is no graph all of whose nodes have different degrees (indeed, the degrees of the n nodes of a simple connected graph are integers between 1 and $n - 1$ included).

Our aim is to establish tradeoffs between the allocated time and the amount of information that has to be given *a priori* to the nodes to enable them to perform leader election. Following the framework of *algorithms with advice*, see, e.g., [10, 13, 18, 38], this information (a single binary string) is provided to all nodes at the start by an oracle knowing the entire network. The length of this string is called the *size of advice*. It should be noted that, since the advice given to all nodes is the same, this information does not increase the asymmetries of the network (unlike in the case when different pieces of information could be given to different nodes) but only helps to harvest the existing asymmetries and use them to elect the leader. Hence the high-level formulation of our problem is the following. What is the minimum amount of identical information that can be given to nodes to enable them to use asymmetries present in the graph to elect a leader in a given time?

Of course, since the faithful map of the network is the total information about it, asking about the minimum size of advice to solve leader election in time τ is meaningful only in the class of networks G for which $\phi(G) \leq \tau$, because otherwise, no advice can help. The central problem of this paper can be now precisely formulated as follows.

> For a given time τ, what is the minimum size
> of advice that permits leader election in time τ,
> for all networks G for which $\phi(G) \leq \tau$?

The paradigm of algorithms with advice has been proven very important in the domain of network algorithms. Establishing a strong lower bound on the minimum size of advice sufficient to accomplish a given task in a given time permits to rule out entire classes of algorithms and thus focus only on possible candidates. For example, if we prove that $\Omega(n/\log n)$ bits of advice are needed to perform a certain task in n-node networks (as we do in this paper for leader election in minimum possible time), this rules out all potential algorithms that can work using only the size n of the network, as n can be given to the nodes using $O(\log n)$ bits. Lower bounds on the size of advice give us impossibility results based strictly on the *amount* of initial knowledge allowed in a model. This is much more general than the traditional approach based on specific categories of information given to nodes, such as the size, diameter, or maximum node degree.

Our results. For a given time τ allocated to leader election, we give upper and lower bounds on the minimum size of advice sufficient to perform leader election in time τ for networks with election index at most α. An upper bound U for a class of networks C means that, for all networks in C, leader election in time τ is possible given advice of size $O(U)$. We prove such a bound by constructing advice of size $O(U)$ together with a leader election algorithm for all networks in the class C, that uses this advice and works in time τ. A lower bound L for a class of networks C means that there exist networks in C for which leader election in time τ requires advice of size $\Omega(L)$. We prove such a bound by constructing a subclass C' of C such that no leader election algorithm running in time τ with advice of size $o(L)$ can succeed for all networks in C'.

We focus on the two sides of the time spectrum. For the smallest possible time, which is the election index of the network, we show that the minimum size of advice is linear in the size n of the network, up to polylogarithmic factors. More precisely, we establish a general upper bound $O(n \log n)$ and lower bounds $\Omega(n \log \log n)$ and $\Omega(n(\log \log n)^2/\log n)$, for election index equal to 1 and larger than 1, respectively.

On the other hand, we consider large values of time: those exceeding the diameter D by a summand, respectively, linear, polynomial, and exponential in the election index; for these values, we prove tight bounds on the minimum size of advice, up to multiplicative constants. More precisely, for any positive integer α, consider the class of networks with election index at most α. Let $c > 1$ be an integer constant. For any graph of election index $\phi \leq \alpha$ in this class, consider leader election algorithms working in time, respectively, at most $D + \phi + c$, at most $D + c\phi$, at most $D + \phi^c$, and at most $D + c^{\phi}$. Hence the additive offset above D in the time of leader election is asymptotically equal to ϕ in the first case, it is linear in ϕ but with a multiplicative constant larger than 1 in the second case, it is polynomial in ϕ but super-linear in the third case, and it is exponential in ϕ in the fourth case. In the considered class we show that the minimum size of advice is $\Theta(\log \alpha)$ in the first case, it is $\Theta(\log \log \alpha)$ in the second case, it is $\Theta(\log \log \log \alpha)$ in the third case, and it is $\Theta(\log(\log^* \alpha))$ in the fourth case. Hence, perhaps surprisingly, the jumps in the minimum size of advice, when the time of leader election varies between the above milestones, are all exponential. We also show that constant advice is not sufficient for leader election in all graphs, regardless of the allocated time.

Related work. The first papers on leader election focused on the scenario where all nodes have distinct labels. Initially, it was investigated for rings in the message passing model. A synchronous algorithm based on label comparisons was given in [28]. It used $O(n \log n)$ messages. An asynchronous algorithm using $O(n \log n)$ messages was given, e.g., in [40]. Leader election was also investigated in the radio communication model, both in the deterministic [30, 33, 37] and in the randomized [42] scenarios.

Many authors [3–6, 43, 44] studied leader election in anonymous networks. In particular, [6, 44] characterize message-passing networks in which leader election is feasible. In [43], the authors study the problem of leader election in general networks, under the assumption that node labels exist but are not unique. They characterize networks in which leader election can be performed and give an algorithm which achieves election when it is feasible. In [12, 14], the authors study message complexity of leader election in rings with possibly nonunique labels. In [11], the authors investigate the feasibility of leader election among anonymous agents that navigate in a network in an asynchronous way.

Providing nodes or agents with arbitrary types of knowledge that can be used to increase efficiency of solutions to network problems has previously been proposed in [1, 10, 13, 16–18, 31, 32, 36, 38, 41]. This approach was referred to as *algorithms with advice*. The advice is given either to the nodes of the network or to mobile agents performing some task in a network. In the first case, instead of advice, the term *informative labeling schemes* is sometimes used if (unlike in our scenario) different nodes can get different information.

Several authors studied the minimum size of advice required to solve network problems in an efficient way. In [32], given a distributed representation of a solution for a problem, the authors investigated the number of bits of communication needed to verify the legality of the represented solution. In [16], the authors compared the minimum size of advice required to solve two information dissemination problems using a linear number of messages. In [18], it was shown that advice of constant size given to the nodes enables the distributed construction of a minimum spanning tree in logarithmic time. In [13], the advice paradigm was used for online problems. In the case of [38], the issue was not efficiency but feasibility: it was shown that $\Theta(n \log n)$ is the minimum size of advice required to perform monotone connected graph clearing. In [10], the task of drawing an isomorphic map by an agent in a graph was considered, and the problem was to determine the minimum advice that has to be given to the agent for the task to be feasible.

Among papers studying the impact of information on the time of leader election, the papers [21, 25, 36] are closest to the present work. In [36], the authors investigated the minimum size of advice sufficient to find the largest-labelled node in a graph, all of whose nodes have distinct labels. The main difference between [36] and the present paper is that we consider networks without node labels. This is a fundamental difference: breaking symmetry in anonymous networks relies heavily on the structure of the graph, rather than on labels, and, as far as results are concerned, much more advice is needed for a given allocated time. In [21], the authors investigated the time of leader election in anonymous networks by characterizing this time in terms of the network size, the diameter of the network, and an additional parameter called the level of symmetry, similar to our election index. This paper used the traditional approach of providing nodes with some parameters of the network, rather than any type of advice, as in our setting. Finally, the paper [25] studied leader election under the advice paradigm for anonymous networks, but restricted attention to trees. It should be stressed that leader election in anonymous trees and in arbitrary anonymous networks present completely different difficulties. The most striking difference is that, in the case of trees, for the relatively modest time equal to the diameter D, leader election can be done in feasible trees without any advice, as all nodes can reconstruct the map of the tree. This should be contrasted with the class of arbitrary networks, in which leader election with no advice is impossible. Our results for large election time values (exceeding the diameter D) give a hierarchy of sharply differing tight bounds on the size of advice in situations in which leader election in trees can be performed with no advice at all.

2 PRELIMINARIES

We use the word "graph" to mean a simple undirected connected graph with unlabeled nodes and all port numbers fixed. In the sequel we use the word "graph" instead of "network".

We will use the following characterization of the election index.

Proposition 2.1. *The election index of a feasible graph is equal to the smallest integer ℓ, such that the augmented truncated views at depth ℓ of all nodes are distinct.*

The value of the election index is estimated in the following proposition, which is an immediate consequence of the main result of [27].

Proposition 2.2. *For any n-node feasible graph of diameter D, its election index is in $O(D \log(n/D))$.*

3 ELECTION IN MINIMUM TIME

We start this section by designing a leader election algorithm working in time ϕ, for any graph of size n and election index ϕ, and using advice of size $O(n \log n)$. The high-level idea of the algorithm is the following. The oracle knowing the graph G produces the advice consisting of three items: the integer ϕ, A_1 and A_2. This is done using Algorithm ComputeAdvice(G). The integer ϕ serves the nodes to determine for how long they have to exchange information with their neighbors. The item A_1 is the most difficult to construct. Its aim is to allow every node that knows its augmented truncated view at depth ϕ (which is acquired in the allocated time ϕ) to construct a unique integer label from the set $\{1, 2, \ldots, n\}$. Recall that the advice is the same for all nodes, and hence each node has to produce a distinct label using this common advice, relying only on its (unique) augmented truncated view at depth ϕ. The third item in the advice, that we call A_2, is a labeled BFS tree of the graph G. (To avoid ambiguity, we take the *canonical* BFS tree, in which the parent of each node u at level $i + 1$ is the node at level i corresponding to the smallest port number at u.) The labels of nodes are equal to those that nodes will construct using item A_1, the root is the node labeled 1, and all port numbers in the BFS tree (that come from the graph G) are faithfully given. More precisely, A_2 is the *code* of this tree, i.e., a binary string of length $O(n \log n)$ which permits the nodes to reconstruct unambiguously this labeled tree (the details are given below). After receiving the entire advice, Algorithm Elect works as follows. Each node acquires its augmented truncated view at depth ϕ, then positions itself in the obtained BFS tree, thanks to the unique constructed label, and outputs the sequence of port numbers corresponding to the unique path from itself to the root of this BFS tree.

The main difficulty is to produce item A_1 of the advice succinctly, i.e., using only $O(n \log n)$ bits, and in such a way that allows nodes to construct unique *short* labels. Note that a naive way in which nodes could attribute themselves distinct labels would require no advice at all and could be done as follows. Nodes could list all possible augmented truncated views at depth ϕ, order them lexicographically in a canonical way, and then each node could adopt as its label the rank in this list. However, already for $\phi = 1$, there are $\Omega(n)^{\Omega(n)}$ different possible augmented truncated views at depth 1, and hence these labels would be of size $\Omega(n \log n)$. Now item A_2 of the advice would have to give the tree with all these labels, thus potentially requiring at least $\Omega(n^2 \log n)$ bits, which significantly exceeds the size of advice that we want to achieve. This is why item A_1 of the advice is needed, and must be constructed in a subtle way. On the one hand, it must be sufficiently short (use only $O(n \log n)$ bits) and on the other hand it must allow nodes to construct distinct labels of size $O(\log n)$. Then item A_2 of the advice can be given using also only $O(n \log n)$ bits.

We now give some intuitions concerning the construction of item A_1 of the advice. This item can be viewed as a carefully constructed *trie*, cf. [2], which is a rooted binary tree whose leaves correspond to objects, and whose internal nodes correspond to yes/no queries concerning these objects. The left child of each internal node corresponds to port 0 and to the answer "no" to the query, and the right child corresponds to port 1 and to the answer "yes" to the query. The object in a given leaf corresponds to all answers on the branch from the root to the leaf, and must be unique. In our case, objects in leaves of the trie are nodes of the graph, and queries serve to discriminate all views $\mathcal{B}^{\phi}(v)$, for all nodes v of the graph G. Since each node v knows its augmented truncated view $\mathcal{B}^{\phi}(v)$, after learning the trie it can position itself as a leaf of it and adopt a unique label from the set $\{1, 2 \ldots, n\}$.

As an example, consider the case $\phi = 1$. All augmented truncated views at depth 1 can be coded by binary sequences of length $O(n \log n)$. In this case the queries at internal nodes of the trie are of two types: "Is the binary representation of your augmented truncated view at depth one of length smaller than t?" (this query is coded as $(0, t)$), and "Is the jth bit of the binary representation of your augmented truncated view at depth one equal to 1?" (this query is coded as $(1, j)$). Since both the possible lengths t and the possible indices j are of size $O(\log n)$, the entire trie can be coded as a binary sequence of length $O(n \log n)$, because there are n leaves of the trie.

For $\phi > 1$ the construction is more complicated. Applying the same method as for $\phi = 1$ (by building a large trie discriminating between all augmented truncated views at depth ϕ, using similar questions as above, only concerning depth ϕ instead of depth 1) is impossible, because the sizes of the queries would exceed $\Theta(\log n)$. Actually, queries would be of size $\Omega(\phi \log n)$, resulting in advice of size $\Omega(\phi n \log n)$, and not $O(n \log n)$. Hence we apply a more subtle strategy. The upper part of the trie is as for the case $\phi = 1$. However, this is not sufficient, as in this case there exist nodes u and v in the graph such that $\mathcal{B}^1(u) = \mathcal{B}^1(v)$ but $\mathcal{B}^{\phi}(u) \neq \mathcal{B}^{\phi}(v)$, and hence such a small trie would not discriminate between all augmented truncated views at depth ϕ. Hence leaves of this partial trie, corresponding to sets of nodes in the graph that have the same augmented truncated view at depth 1, have to be further developed, by adding sub-tries rooted at these leaves, to further discriminate between all augmented truncated views at depth ϕ. This is done recursively in such a way that these further queries are still of size $O(\log n)$, and constitutes the main difficulty of the advice construction.

The details of Algorithm ComputeAdvice and of Algorithm Elect using it are in the Appendix.

The next theorem shows that Algorithm Elect, executed on any n-node graph, performs leader election in time equal to the election index of this graph, using advice of size $O(n \log n)$.

Theorem 3.1. *For any n-node graph G with election index ϕ, the following properties hold:*

(1) *Algorithm* ComputeAdvice(G) *terminates and returns a binary string of length $O(n \log n)$.*

(2) *Using the advice returned by Algorithm* `ComputeAdvice(G)`, *Algorithm* `Elect` *performs leader election in time* ϕ.

We now prove two lower bounds on the size of advice for election in the minimum time, i.e., in time equal to the election index ϕ. For $\phi = 1$ we establish the lower bound $\Omega(n \log\log n)$, and for $\phi > 1$ we establish the lower bound $\Omega(n(\log\log n)^2/\log n)$. Both these bounds differ from our upper bound $O(n \log n)$ only by a polylogarithmic factor.

The high-level idea of the proofs of these bounds is the following. Given a positive integer ϕ we construct, for arbitrarily large integers n, families of n-node graphs with election index ϕ and with the property that each graph of such a family must receive different advice for any election algorithm working in time ϕ for all graphs of this family. This property is established by showing that, if two graphs G_1 and G_2 from the family received the same advice, some nodes v_1 in G_1 and v_2 in G_2 would have to output identical sequences of port numbers because they have identical augmented truncated views at depth ϕ, which would result in failure of leader election in one of these graphs. Since the constructed families are large enough, the above property implies the desired lower bound on the size of advice for at least one graph in the family.

We start with the construction of a family $\mathcal{F}(x) = \{C_1, \ldots, C_y\}$ of labeled $(x+1)$-node cliques, for $x \geq 2$. All these cliques will have node labels $r, v_0, v_1, \ldots, v_{x-1}$. We first define a clique C by assigning its port numbers. Assign port number i, for $0 \leq i \leq x-1$, to the port at r corresponding to the edge $\{r, v_i\}$. The rest of the port numbers are assigned arbitrarily. We now show how to produce the cliques of the family $\mathcal{F}(x)$ from the clique C. Consider all sequences of x integers from the set $\{1, \ldots, x-1\}$. There are $y = (x-1)^x$ such sequences. Let $(s_1, \ldots, s_y)$ be any enumeration of them. Let $s_t = (h_0, h_1, \ldots, h_{x-1})$, for a fixed $t = 1, \ldots, y$. The clique C_t is defined from clique C by assigning port $(p + h_j) \mod x$ instead of port p at node v_j, for all pairs $0 \leq j, p \leq x-1$.

We first consider the election index $\phi = 1$.

Theorem 3.2. *For arbitrarily large integers n, there exist n-node graphs with election index 1, such that leader election in time 1 in these graphs requires advice of size* $\Omega(n \log\log n)$.

Proof. (Sketch) Fix an integer $k \geq 2^{16}$, and let $x = \lceil 2\log k/\log\log k \rceil$. We have $k \leq y = (x-1)^x$. We first define a graph H_k using the family $\mathcal{F}(x)$, cf. Fig. 1.

Consider a ring of size k with nodes $w_1, \ldots, w_k$. Attach an isomorphic copy of the clique C_t to node w_t, by identifying w_t with node r of this copy and taking all other nodes different from the nodes of the ring. (The term "isomorphic" means that all port numbers are preserved.) All attached cliques are pairwise node-disjoint. Assign ports x and $x+1$ corresponding to edges of the ring at each of its nodes, in the clockwise order. This concludes the construction of graph H_k.

Finally we produce a family $\mathcal{G}_k$ consisting of $(k-1)!$ graphs as follows. Keep the clique at node w_1 of the ring in H_k fixed, and permute arbitrarily cliques attached to all other nodes of the ring. Then delete all node labels.

The proof relies on the following two claims, proved in the Appendix. The first claim establishes the election index of graphs in $\mathcal{G}_k$.

Claim 3.1. *All graphs in the family* $\mathcal{G}_k$ *have election index 1.*

The next claim will imply a lower bound on the number of different pieces of advice needed to perform election in the family $\mathcal{G}_k$ in time 1.

Claim 3.2. *Consider any election algorithm working for the family* $\mathcal{G}_k$ *in time 1. The advice given to distinct graphs in this family must be different.*

Our lower bound will be shown on the family $\mathcal{G} = \bigcup_{k=2^{16}}^{\infty} \mathcal{G}_k$. Consider an election algorithm working in all graphs of this family in time 1. For any $k \geq 2^{16}$, let $n_k = k(\lceil 2\log k/\log\log k \rceil + 1)$. Graphs in the family $\mathcal{G}_k$ have size n_k. By Claim 3.1, all these graphs have election index 1. By Claim 3.2, all of them must get different advice. Since, for any $k \geq 2^{16}$, there are $(k-1)!$ graphs in $\mathcal{G}_k$, at least one of them must get advice of size $\Omega(\log((k-1)!)) = \Omega(k \log k)$. We have $k \log k \in \Theta(n_k \log\log n_k)$. Hence there exists an infinite sequence of integers n_k such that there are n_k-node graphs with election index 1 that require advice of size $\Omega(n_k \log\log n_k)$ for election in time 1. □

We next consider the election index $\phi > 1$. The lower bound in this case uses a construction slightly more complicated than for $\phi = 1$. Note that a straightforward generalization of the previous construction would lead to a lower bound $\Omega(n \log\log n/\phi)$ which would be too weak for our purpose, as ϕ can be much larger than polylogarithmic in n.

Theorem 3.3. *Let* ϕ *be an integer larger than 1. For arbitrarily large integers n, there exist n-node graphs with election index* ϕ, *such that leader election in time* ϕ *in these graphs requires advice of size* $\Omega(n(\log\log n)^2/\log n)$.

4 ELECTION IN LARGE TIME

In this section we study the minimum size of advice sufficient to perform leader election when the allocated time is large, i.e., when it exceeds the diameter of the graph by an additive offset which is some function of the election index ϕ of the graph. We consider four values of this offset, for an integer constant $c > 1$: $\phi + c$, $c\phi$, ϕ^c, and c^{ϕ}. In the first case the offset is asymptotically equal to ϕ, in the second case it is linear in ϕ but the multiplicative constant is larger than 1, in the third case it is polynomial in ϕ but super-linear, and in the fourth case it is exponential in ϕ. Note that, even in the first case, that calls for the fastest election among these four cases (in time $D + \phi + c$), the allocated time is large enough for all nodes to see all the differences in truncated views of other nodes, which makes a huge difference between leader election in such a time and in the minimum possible time ϕ. For all these four election times, we establish tight bounds on the minimum size of advice that enables election in this time, up to multiplicative constants.

We start by designing an algorithm that performs leader election in time at most $D + x + 1$, for any graph of diameter D and election index ϕ, provided that nodes receive as input an integer $x \geq \phi$. Note that nodes of the graph know neither D nor ϕ. We will then show how to derive from this generic algorithm four leader election algorithms using larger and larger time and smaller and smaller advice.

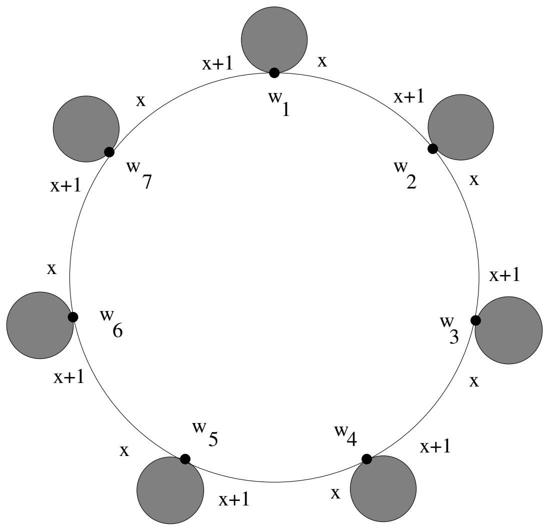


Figure 1: A representation of the graph H_7. The grey discs represent cliques from $\mathcal{F}(x)$.

The high-level idea of Algorithm Generic(x) is the following. Nodes of the graph communicate between them and acquire augmented truncated views at increasing depths. Starting from round x, they discover augmented truncated views at depth x of an increasing set of nodes. They stop in the round when no new augmented truncated views at depth x are discovered. At this time, we have the guarantee that all nodes learned all augmented truncated views at depth x. Hence, to solve leader election, it suffices that every node outputs a sequence of port numbers leading to a node with the lexicographically smallest augmented truncated view at depth x. Since $x \geq \phi$, the augmented truncated view at depth x of every node is unique in the graph, in view of Proposition 2.1. Hence all nodes output sequences of port numbers leading to the same node.

The detailed description of Algorithm Generic is in the Appendix. The following lemma establishes the correctness of Algorithm Generic and estimates its execution time.

Lemma 4.1. *For any graph G of diameter D and election index ϕ, Algorithm* Generic(x)*, with any parameter $x \geq \phi$, is a correct leader election algorithm and works in time at most $D + x + 1$.*

We now describe four algorithms, called Election$_i$, for $i = 1, 2, 3, 4$, working for graphs of diameter D and election index ϕ. Recall the notation ${}^{i}c$ defined by induction as follows: ${}^{0}c = 1$ and ${}^{i+1}c = c^{{}^{i}c}$. Intuitively it denotes a tower of powers. For an integer constant $c > 1$, let $T_1 = D+\phi+c$, $T_2 = D+c\phi$, $T_3 = D+\phi^c$, and $T_4 = D+c^{\phi}$. Let A_1 be the binary representation of ϕ, let A_2 be the binary representation of $\lfloor \log \phi \rfloor$, let A_3 be the binary representation of $\lfloor \log \log \phi \rfloor$, and let A_4 be the binary representation of $\log^* \phi$. Hence the size of A_1 is $O(\log \phi)$, the size of A_2 is $O(\log \log \phi)$, the size of A_3 is $O(\log \log \log \phi)$, and the size of A_4 is $O(\log(\log^* \phi))$. Define the following integers. $P_1 = \phi$, $P_2 = 2^{\lfloor \log \phi \rfloor + 1} - 1$, $P_3 = 2^{2^{\lfloor \log \log \phi \rfloor + 1}} - 1$, and $P_4 = {}^{(\log^* \phi)+1}2 - 1$.

Algorithm Election$_i$ uses advice A_i and will be shown to work in time T_i:

```
Algorithm 1 Election_i
    Generic($P_i$)
```

Theorem 4.2. *Let G be a graph of diameter D and election index ϕ. Let $c > 1$ be any integer constant.*

(1) *Algorithm* Election$_1$ *solves leader election in G in time at most $D + \phi + c$ and using $O(\log \phi)$ bits of advice.*
(2) *Algorithm* Election$_2$ *solves leader election in G in time at most $D + c\phi$ and using $O(\log \log \phi)$ bits of advice.*
(3) *Algorithm* Election$_3$ *solves leader election in G in time at most $D + \phi^c$ and using $O(\log \log \log \phi)$ bits of advice.*
(4) *Algorithm* Election$_4$ *solves leader election in G in time at most $D + c^{\phi}$ and using $O(\log(\log^* \phi))$ bits of advice.*

Proof. 1. Algorithm Election$_1$ uses advice A_1 which is the binary representation of ϕ of size $O(\log \phi)$. It first computes ϕ using A_1, and then calls Algorithm Generic(ϕ). In view of Lemma 4.1, Algorithm Generic(ϕ) solves leader election in G in time at most $D + \phi + 1 \leq D + \phi + c$. This proves part 1 of the theorem.

2. Algorithm Election$_2$ uses advice A_2 which is the binary representation of $\lfloor \log \phi \rfloor$ of size $O(\log \log \phi)$. It first computes $\lfloor \log \phi \rfloor$ using A_2, then computes $P_2 = 2^{\lfloor \log \phi \rfloor + 1} - 1$ and calls Algorithm Generic(P_2). Notice that $P_2 \geq \phi$. In view of Lemma 4.1, Algorithm Generic(P_2) solves leader election in G in time at most $D + P_2 + 1 = D + 2^{\lfloor \log \phi \rfloor + 1}$. This is at most $D + 2\phi$ and hence at most $D + c\phi$, since c is an integer larger than 1.

3. Algorithm Election$_3$ uses advice A_3 which is the binary representation of $\lfloor \log \log \phi \rfloor$ of size $O(\log \log \log \phi)$. It first computes $\lfloor \log \log \phi \rfloor$ using A_3, then computes $P_3 = 2^{2^{\lfloor \log \log \phi \rfloor + 1}} - 1$ and calls Algorithm Generic(P_3). Notice that $P_3 \geq \phi$. In view of Lemma 4.1, Algorithm Generic(P_3) solves leader election in G in time at most $D + P_3 + 1 = D + 2^{2^{\lfloor \log \log \phi \rfloor + 1}}$. This is at most $D + \phi^2$ and hence at most $D + \phi^c$, since c is an integer larger than 1.

4. Algorithm $\texttt{Election}_4$ uses advice A_4 which is the binary representation of $\log^* \phi$ of size $O(\log(\log^* \phi))$. It first computes $\log^* \phi$ using A_4, then computes $P_4 = {}^{(\log^* \phi)+1}2 - 1$ and calls Algorithm $\texttt{Generic}(P_4)$. Notice that $P_4 \geq \phi$. In view of Lemma 4.1, Algorithm $\texttt{Generic}(P_4)$ solves leader election in G in time at most $D + P_4 + 1 = D + {}^{(\log^* \phi)+1}2$. This is at most $D + 2^\phi$ and hence at most $D + c^\phi$, since c is an integer larger than 1. □

Remark. Notice that the first part of the theorem remains valid for $c = 1$, and the proof remains the same. Hence, it is possible to perform leader election in time $D+\phi+1$ using $O(\log \phi)$ bits of advice. We do not know if the same is true for the time $D+\phi$, but in this time it is possible to elect a leader using $O(\log D + \log \phi)$ bits of advice. Indeed, it suffices to provide the nodes with values of the diameter D and of the election index ϕ. Equipped with this information, each node u learns $\mathcal{B}^{D+\phi}(u)$ in time $D + \phi$. Then, knowing D, it knows that nodes that it sees in this augmented truncated view at distance at most D from the root of this view represent all nodes of the graph. Knowing the value of ϕ, node u can reconstruct $\mathcal{B}^{\phi}(v)$, for each such node v, and hence find in $\mathcal{B}^{D+\phi}(u)$ a representation of the node w in the graph, whose augmented truncated view $\mathcal{B}^{\phi}(v)$ is lexicographically smallest. Finally, the node u can output a sequence of port numbers corresponding to one of the shortest paths from u to w in $\mathcal{B}^{D+\phi}(u)$.

The following theorem provides matching lower bounds (up to multiplicative constants) on the minimum size of advice sufficient to perform leader election in time corresponding to our four milestones.

Theorem 4.3. *Let α be a positive integer, and let $c > 1$ be any integer constant.*

(1) *Consider any leader election algorithm such that for all positive integers D and ϕ, this algorithm works in time at most $D + \phi + c$, for all graphs of diameter D and election index ϕ. There exist graphs with election index at most α such that this algorithm in these graphs requires advice of size $\Omega(\log \alpha)$.*
(2) *Consider any leader election algorithm such that for all positive integers D and ϕ, this algorithm works in time at most $D + c\phi$, for all graphs of diameter D and election index ϕ. There exist graphs with election index at most α such that this algorithm in these graphs requires advice of size $\Omega(\log \log \alpha)$.*
(3) *Consider any leader election algorithm such that for all positive integers D and ϕ, this algorithm works in time at most $D + \phi^c$, for all graphs of diameter D and election index ϕ. There exist graphs with election index at most α such that this algorithm in these graphs requires advice of size $\Omega(\log \log \log \alpha)$.*
(4) *Consider any leader election algorithm such that for all positive integers D and ϕ, this algorithm works in time at most $D + c^\phi$, for all graphs of diameter D and election index ϕ. There exist graphs with election index at most α such that this algorithm in these graphs requires advice of size $\Omega(\log(\log^* \alpha))$.*

Proof. (Sketch) The high-level idea of the proof relies on the construction of (ordered) families of graphs with controlled growth of election indices, and with the property that, for any election algorithm working in the prescribed time, graphs from different families must receive different advice. Since the growth of election indices in the constructed sequence of families is controlled (it is linear in part 1), this implies the desired lower bound on the size of advice. In order to prove that advice in different families must be different, we have to show that otherwise the algorithm would fail in one of these families. The difficulty lies in constructing the families of graphs in such a way as to confuse the hypothetical algorithm in spite of the fact that, as opposed to the situation in Theorems 3.2 and 3.3, the algorithm has now a lot of time: nodes can see all the differences in augmented truncated views of other nodes. In this situation, the way to confuse the algorithm is to make believe two nodes that they are in a graph with a smaller diameter and thus have to stop early, outputting a path to the leader, while in reality they are in a graph of large diameter, and each of them has seen less than "half" of the graph, which results in outputting by each of them a path to a different leader. These nodes are fooled because their augmented truncated views in the large graph at the depth requiring them to stop in the smaller graph are the same as in this smaller graph. In order to assure this, parts of the smaller graph must be carefully reproduced in the large graph, which significantly complicates the construction and the analysis. □

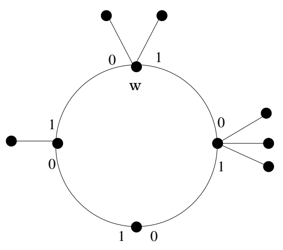
We close this section by showing that constant advice is not enough for leader election in all feasible graphs, regardless of the allocated time.

Proposition 4.4. *There is no algorithm using advice of constant size and performing leader election in all feasible graphs.*

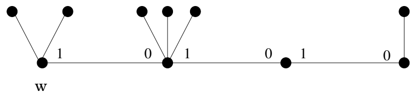
Proof. We define a family $\mathcal{H}$ of graphs, called *hairy rings*, for which we will prove that no algorithm with advice of constant size performs correct leader election for all graphs in $\mathcal{H}$. Let R_n be the ring of size $n \geq 3$, with port numbers 0,1 at each node, in clockwise order. Let S_k, for any integer $k \geq 2$, be the $(k + 1)$-node tree with k leaves, called the k-star.

The only node of degree larger than 1 of the k-star is called its *central node*. For $k = 1$, S_k is defined as the two-node graph with the central node designated arbitrarily, and for $k = 0$, S_k is defined as the one-node graph, with the unique node being its central node. The class $\mathcal{H}$ is the set of all graphs that can be obtained in the following way. For all $n \geq 3$, attach to every node v of every ring R_n some graph S_k, for $k \geq 0$, by identifying its central node with the node v, in such a way that, for every ring, the star of maximum size attached to it is unique. Assign missing port numbers in any legal way, i.e., so that port numbers at a node of degree d are from 0 to $d - 1$. Every graph obtained in this way is feasible because it has a unique node of maximum degree. An example of a hairy ring is depicted in Fig. 2a.

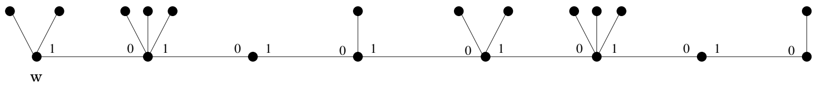
For any graph H in $\mathcal{H}$ we define a *cut* of H as follows, see Fig. 2b. Let H be a graph resulting from a ring R_n by attaching stars. Fix any node $w = w_1$ of this ring. Let $w_1, \ldots, w_n$ be nodes of this ring listed in clockwise order. The cut of H at node w is the graph resulting from H by removing the edge $\{w_1, w_n\}$. Node $w = w_1$ is called the first node of the cut and node w_n is called the last node of the cut. For any integer $\gamma \geq 2$, the γ-*stretch* of H starting at node w is the graph defined as follows, see Fig. 2c. Take γ pairwise disjoint



(a) a hairy ring H.



(b) the cut of H at node w.



(c) the 2-stretch of H at node w.

Figure 2: Illustration of a hairy ring and its different transformations used in the proof of Proposition 4.4.

isomorphic copies of the cut of H at w. For $1 < i \leq \gamma$, attach the ith copy to the $(i-1)$th copy joining the first node a_i of the ith copy with the last node b_{i-1} of the $(i-1)$th copy by an edge with port 0 at a_i and port 1 at b_{i-1} . The first node of the first copy is called the first node of the γ-stretch, and the last node of the last copy is called the last node of the γ-stretch.

Suppose that there exists a leader election algorithm $\mathcal{A}$ which uses advice of constant size to perform leader election in all hairy rings from the family $\mathcal{H}$. Let c be the smallest integer such that a total of c pieces of advice are sufficient to elect a leader in every graph from $\mathcal{H}$ by algorithm $\mathcal{A}$. Let $H_1, \ldots, H_c$ be graphs from $\mathcal{H}$ for which algorithm $\mathcal{A}$ uses different pieces of advice. Let N be the maximum of sizes of all graphs $H_1, \ldots, H_c$, and let T be the maximum execution time of $\mathcal{A}$, for all graphs $H_1, \ldots, H_c$.

Let $\gamma = 4(N+T)$ and let G_j be the γ-stretch of H_j starting at some node u_j of H_j, for $j \leq c$. We define the graph G as follows. Take pairwise disjoint isomorphic copies of graphs G_j, for $j \leq c$. For every $1 < j \leq c$, attach G_j to G_{j-1} joining the first node c_i of the ith copy with the last node d_{i-1} of the $(i-1)$th copy by an edge with port 0 at c_i and port 1 at d_{i-1}. Finally, take a γ-star and join its central node by edges to the first node of G_1 and to the last node of G_γ, assigning missing port numbers in any legal way. The graph G obtained in this way is in $\mathcal{H}$ because it has a unique node of maximum degree which is $\gamma+2$. Let n_{H_j} be the size of the ring that was used in the construction of H_j. Let a_j be the (unique) node of G_j at distance $n_{H_j}(N+T)$ from u_j, at the end of a simple path all of whose ports are 0's and 1's. Let b_j be the (unique) node of G_j at distance $3n_{H_j}(N+T)$ from u_j, at the end of a simple path all of whose ports are 0's and 1's. Call these nodes the *foci* of G_j. Each of them corresponds to the first node of the cut serving to define G_j. Let z_j be the node in H_j at which this cut was done.

By definition of graphs $H_1, \ldots, H_c$, the advice received by graph G when algorithm $\mathcal{A}$ is performed, is the same as that received by some graph H_{j_0}. In H_{j_0} the node z_{j_0} executing algorithm $\mathcal{A}$ must stop after time at most T. By construction, the augmented truncated view $\mathcal{B}^T(z_{j_0})$ in H_{j_0} is the same as the augmented truncated views $\mathcal{B}^T(a_{j_0})$ and $\mathcal{B}^T(b_{j_0})$ in G. Hence nodes a_{j_0} and b_{j_0} executing algorithm $\mathcal{A}$ in G must also stop after time at most T. Node z_{j_0} in H_{j_0} must output a sequence of port numbers of length smaller than $2N$ because the size of H_{j_0} is at most N. Hence nodes a_{j_0} and b_{j_0} executing algorithm $\mathcal{A}$ in G must also output a sequence of port numbers of length smaller than $2N$, corresponding to simple paths in G of length smaller than N, starting, respectively at nodes a_{j_0} and b_{j_0}. However, the distance between a_{j_0} and b_{j_0} in G is at least $2N$, hence the other extremities of these simple paths must be different. It follows that the leaders elected by nodes a_{j_0} and b_{j_0} executing algorithm $\mathcal{A}$ in G are different, and hence this algorithm is not correct for the class $\mathcal{H}$. □

5 CONCLUSION

We established almost tight bounds on the minimum size of advice sufficient for election in minimum possible time (i.e., in time equal to the election index ϕ) and tight bounds on this size for several large values of time. The first big jump occurs between time ϕ and time $D+\phi$, where D is the diameter of the graph. In the first case, the size of advice is (roughly) linear in the size n of the graph, and in the second case it is at most logarithmic in n, in view of Proposition 2.2 and of the remark after Theorem 4.2. The intriguing open question left by our results is how the minimum size of advice behaves in the range of election time strictly between ϕ and $D+\phi$, i.e., for time sufficiently large to elect if the map were known, but possibly too small for all nodes to see the augmented truncated views at depth ϕ of all other nodes, and hence to realize all the differences in views. Note that, for time exactly $D+\phi$, all nodes see all these differences, although, without any advice, they cannot realize that they see all of them: this is why some advice is needed for time $D+\phi$.

In this paper we defined leader election in anonymous networks similarly as in [25]: a node has to output a simple path to the leader. There is, however, another natural definition of leader election in this context: it can be argued that it is enough for every node to learn a port corresponding to a shortest path towards the leader, as then, e.g., in the case when all nodes have to send some messages to the leader, packets could be routed to the leader from node to node, using only this local information. This is indeed true, if nodes want to cooperate with others by revealing the local port towards the leader when retransmitting packets. In some applications, however, such a cooperation may be uncertain, and even when it happens, it may slow down transmissions, as the local port has to be retrieved from the memory of the relaying node. Hence, both definitions may be suitable in different applications. It would be interesting to compare the size of advice needed for fast leader election in both these variations.

REFERENCES

[1] S. Abiteboul, H. Kaplan, T. Milo, Compact labeling schemes for ancestor queries, Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), 547–556.

[2] A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley 1983.

[3] D. Angluin, Local and global properties in networks of processors. Proc. 12th Annual ACM Symposium on Theory of Computing (STOC 1980), 82–93.

[4] H. Attiya and M. Snir, Better computing on the anonymous Ring, Journal of Algorithms 12, (1991), 204-238.

[5] H. Attiya, M. Snir, and M. Warmuth, Computing on an anonymous ring, Journal of the ACM 35, (1988), 845-875.

[6] P. Boldi, S. Shammah, S. Vigna, B. Codenotti, P. Gemmell, and J. Simon, Symmetry breaking in anonymous networks: Characterizations. Proc. 4th Israel Symposium on Theory of Computing and Systems, (ISTCS 1996), 16-26.

[7] P. Boldi and S. Vigna, Computing anonymously with arbitrary knowledge, Proc. 18th ACM Symp. on Principles of Distributed Computing (PODC 1999), 181-188.

[8] J.E. Burns, A formal model for message passing systems, Tech. Report TR-91, Computer Science Department, Indiana University, Bloomington, September 1980.

[9] F. Chierichetti, personal communication.

[10] D. Dereniowski, A. Pelc, Drawing maps with advice, Journal of Parallel and Distributed Computing 72 (2012), 132–143.

[11] D. Dereniowski, A. Pelc, Leader election for anonymous asynchronous agents in arbitrary networks, Distributed Computing 27 (2014), 21-38.

[12] S. Dobrev and A. Pelc, Leader election in rings with nonunique labels, Fundamenta Informaticae 59 (2004), 333-347.

[13] Y. Emek, P. Fraigniaud, A. Korman, A. Rosen, Online computation with advice, Theoretical Computer Science 412 (2011), 2642–2656.

[14] P. Flocchini, E. Kranakis, D. Krizanc, F.L. Luccio and N. Santoro, Sorting and election in anonymous asynchronous rings, Journal of Parallel and Distributed Computing 64 (2004), 254-265.

[15] P. Fraigniaud, C. Gavoille, D. Ilcinkas, A. Pelc, Distributed computing with advice: Information sensitivity of graph coloring, Distributed Computing 21 (2009), 395–403.

[16] P. Fraigniaud, D. Ilcinkas, A. Pelc, Communication algorithms with advice, Journal of Computer and System Sciences 76 (2010), 222–232.

[17] P. Fraigniaud, D. Ilcinkas, A. Pelc, Tree exploration with advice, Information and Computation 206 (2008), 1276–1287.

[18] P. Fraigniaud, A. Korman, E. Lebhar, Local MST computation with short advice, Theory of Computing Systems 47 (2010), 920–933.

[19] G.N. Fredrickson and N.A. Lynch, Electing a leader in a synchronous ring, Journal of the ACM 34 (1987), 98-115.

[20] E. Fusco, A. Pelc, How much memory is needed for leader election, Distributed Computing 24 (2011), 65-78.

[21] E. Fusco, A. Pelc, Knowledge, level of symmetry, and time of leader election, Distributed Computing 28 (2015), 221-232.

[22] E. Fusco, A. Pelc, Trade-offs between the size of advice and broadcasting time in trees, Algorithmica 60 (2011), 719–734.

[23] E. Fusco, A. Pelc, R. Petreschi, Use knowledge to learn faster: Topology recognition with advice, Information and Computation 247 (2016), 254-265.

[24] C. Gavoille, D. Peleg, S. Pérennes, R. Raz. Distance labeling in graphs, Journal of Algorithms 53 (2004), 85-112.

[25] C. Glacet, A. Miller, A. Pelc, Time vs. information tradeoffs for leader election in anonymous trees, Proc. 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016), 600-609.

[26] M.A. Haddar, A.H. Kacem, Y. Métivier, M. Mosbah, and M. Jmaiel, Electing a leader in the local computation model using mobile agents. Proc. 6th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2008), 473-480.

[27] J. Hendrickx, Views in a graph: To which depth must equality be checked?, *IEEE Transactions on Parallel and Distributed Systems* 25 (2014) 1907-1912.

[28] D.S. Hirschberg, and J.B. Sinclair, Decentralized extrema-finding in circular configurations of processes, Communications of the ACM 23 (1980), 627-628.

[29] D. Ilcinkas, D. Kowalski, A. Pelc, Fast radio broadcasting with advice, Theoretical Computer Science, 411 (2012), 1544–1557.

[30] T. Jurdzinski, M. Kutylowski, and J. Zatopianski, Efficient algorithms for leader election in radio networks. Proc., 21st ACM Symp. on Principles of Distributed Computing (PODC 2002), 51-57.

[31] M. Katz, N. Katz, A. Korman, D. Peleg, Labeling schemes for flow and connectivity, SIAM Journal of Computing 34 (2004), 23–40.

[32] A. Korman, S. Kutten, D. Peleg, Proof labeling schemes, Distributed Computing 22 (2010), 215–233.

[33] D. Kowalski, and A. Pelc, Leader election in ad hoc radio networks: A keen ear helps, Journal of Computer and System Sciences 79 (2013), 1164-1180.

[34] G. Le Lann, Distributed systems - Towards a formal approach, Proc. IFIP Congress, 1977, 155–160, North Holland.

[35] N.L. Lynch, Distributed Algorithms, Morgan Kaufmann Publ. Inc., San Francisco, USA, 1996.

[36] A. Miller, A. Pelc: Election vs. selection: Two ways of finding the largest node in a graph, Proc. 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2016), 377-386.

[37] K. Nakano and S. Olariu, Uniform leader election protocols for radio networks, IEEE Transactions on Parallel and Distributed Systems 13 (2002), 516-526.

[38] N. Nisse, D. Soguet, Graph searching with advice, Theoretical Computer Science 410 (2009), 1307–1318.

[39] D. Peleg, Distributed Computing, A Locality-Sensitive Approach, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia 2000.

[40] G.L. Peterson, An $O(n \log n)$ unidirectional distributed algorithm for the circular extrema problem, ACM Transactions on Programming Languages and Systems 4 (1982), 758-762.

[41] M. Thorup, U. Zwick, Approximate distance oracles, Journal of the ACM, 52 (2005), 1–24.

[42] D.E. Willard, Log-logarithmic selection resolution protocols in a multiple access channel, SIAM J. on Computing 15 (1986), 468-477.

[43] M. Yamashita and T. Kameda, Electing a leader when procesor identity numbers are not distinct, Proc. 3rd Workshop on Distributed Algorithms (WDAG 1989), LNCS 392, 303-314.

[44] M. Yamashita and T. Kameda, Computing on anonymous networks: Part I - Characterizing the solvable cases, IEEE Trans. Parallel and Distributed Systems 7 (1996), 69-89.